FIG. 1

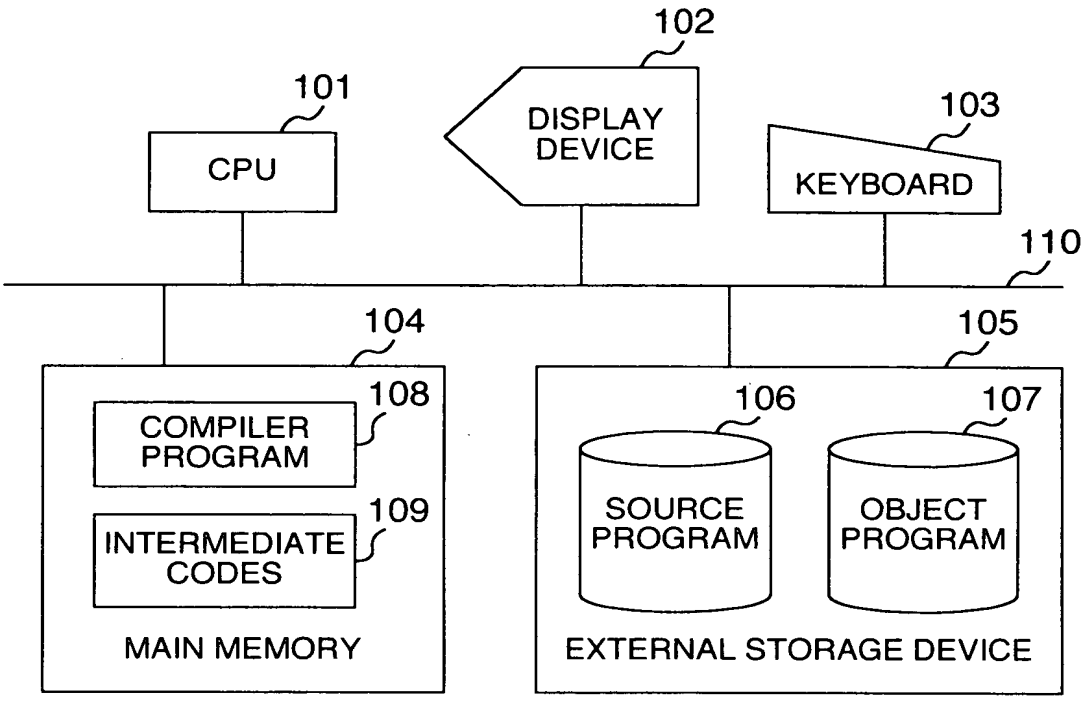

FIG. 2

200

```
while (cond) {          (201)
   c = a+b;             (202)
   *p = c;              (203)
   p = p->next;         (204)
}                       (205)
```

FIG. 3

300

```
while (cond) {              (301)
   ld   r1=[&a]             (302)
   ld   r2=[&b]             (303)
   add  r3=r1,r2            (304)
   st   *p=r3               (305)
   p = p->next              (306)
}                           (307)
```

FIG. 4

400

```
ld.a  r1=[&a]               (401)
ld.a  r2=[&b]               (402)
add   r3=r1,r2              (403)
while (cond) {              (404)
    chk.a  r1,recover1      (405)
L1: chk.a  r2,recover2      (406)
L2: st     *p=r3            (407)
    p = p->next             (408)
}                           (409)
-----------------------------------
recover1:                   (410)
ld.a  r1=[&a]               (411)
add   r3=r1,r2              (412)
br    L1                    (413)
-----------------------------------
recover2:                   (414)
ld.a  r2=[&b]               (415)
add   r3=r1,r2              (416)
br    L2                    (417)
```

FIG. 5

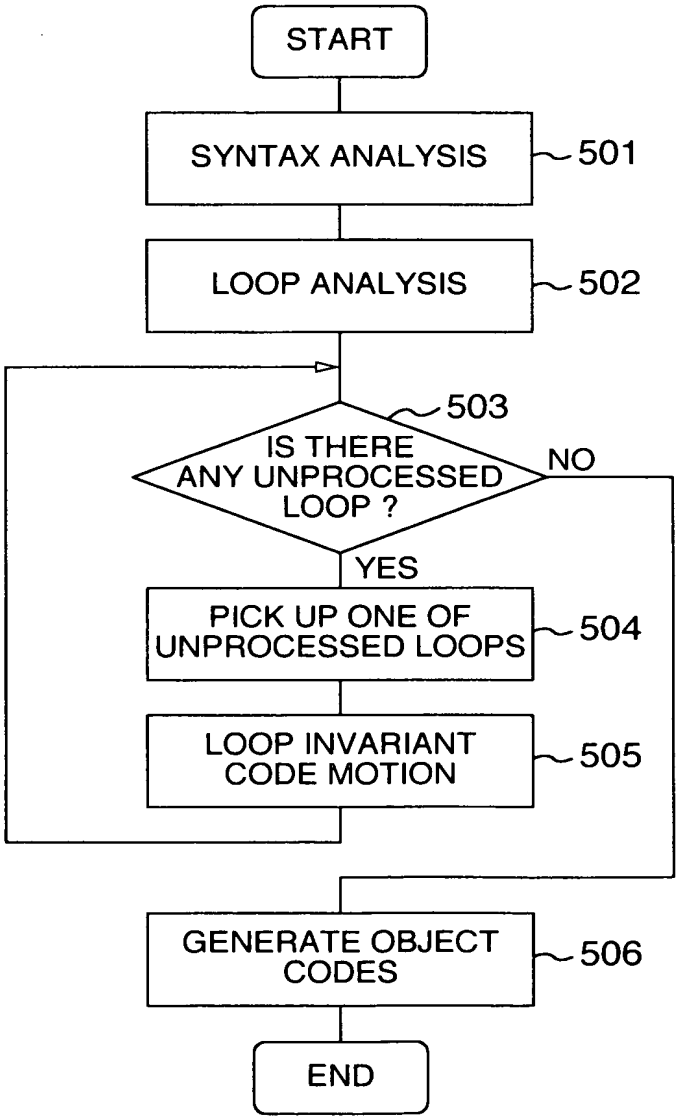
START
SYNTAX ANALYSIS
501
LOOP ANALYSIS
502
503
IS THERE ANY UNPROCESSED LOOP ?
NO
YES
PICK UP ONE OF UNPROCESSED LOOPS
504
LOOP INVARIANT CODE MOTION
505
GENERATE OBJECT CODES
506
END

FIG. 6

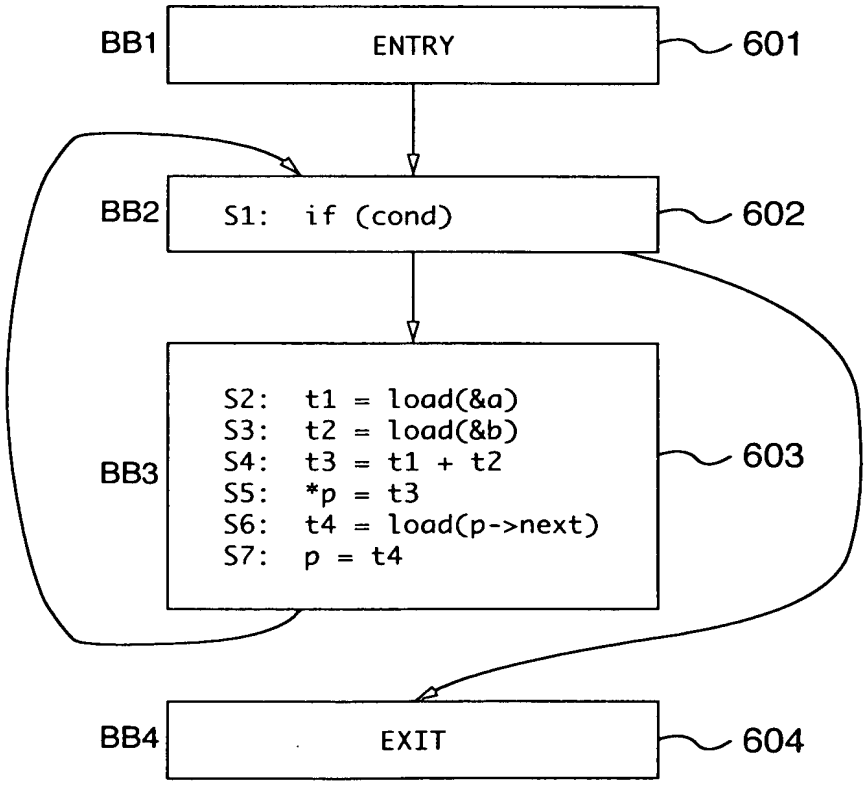
BB1
ENTRY
601
BB2
S1: if (cond)
602
BB3
S2: t1 = load(&a)
S3: t2 = load(&b)
S4: t3 = t1 + t2
S5: *p = t3
S6: t4 = load(p->next)
S7: p = t4
603
BB4
EXIT
604

FIG. 7

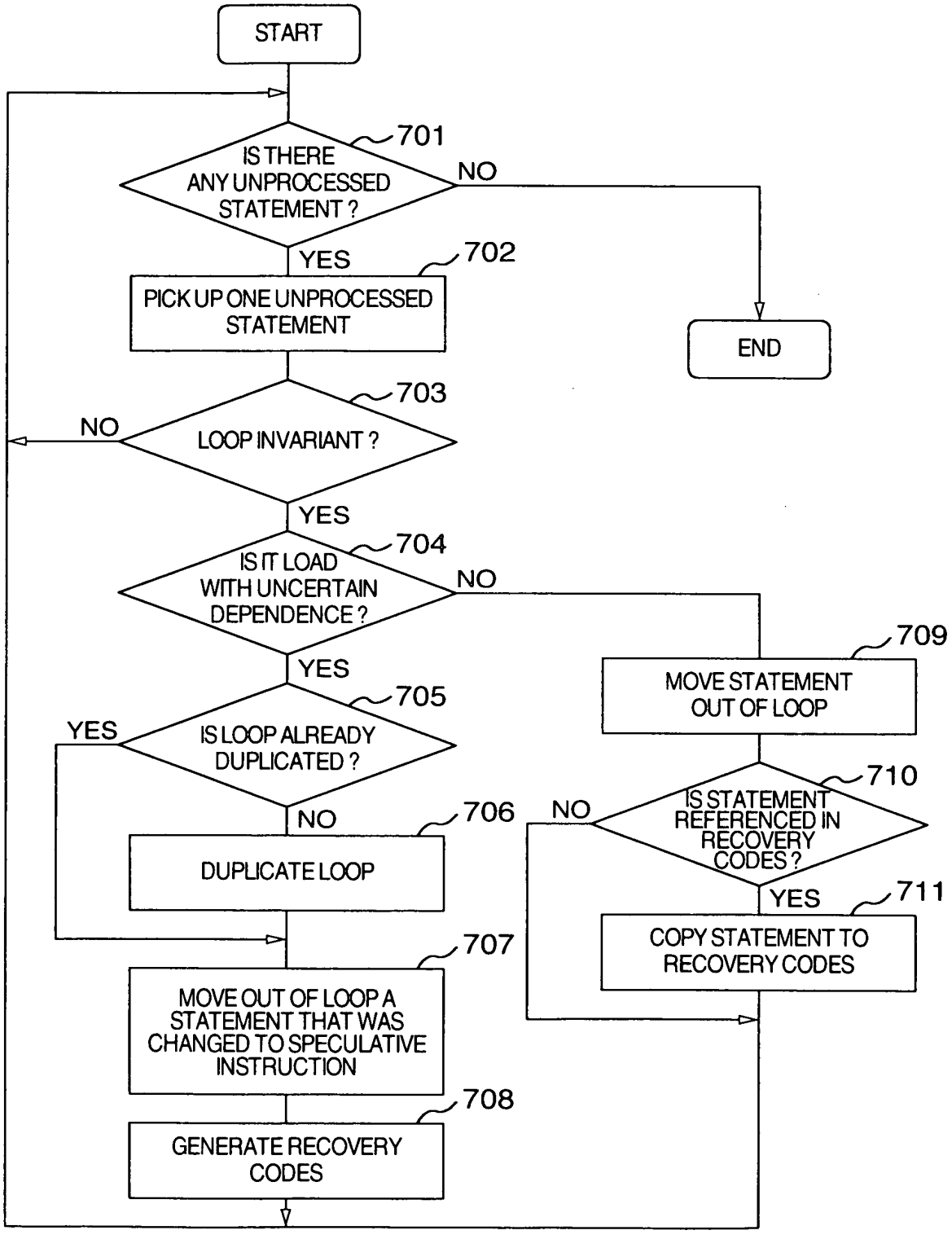
START
701
IS THERE ANY UNPROCESSED STATEMENT ?
NO
YES
702
PICK UP ONE UNPROCESSED STATEMENT
END
703
NO
LOOP INVARIANT ?
YES
704
IS IT LOAD WITH UNCERTAIN DEPENDENCE ?
NO
YES
705
YES
IS LOOP ALREADY DUPLICATED ?
NO
706
DUPLICATE LOOP
707
MOVE OUT OF LOOP A STATEMENT THAT WAS CHANGED TO SPECULATIVE INSTRUCTION
708
GENERATE RECOVERY CODES
709
MOVE STATEMENT OUT OF LOOP
710
NO
IS STATEMENT REFERENCED IN RECOVERY CODES ?
YES
711
COPY STATEMENT TO RECOVERY CODES

FIG. 8

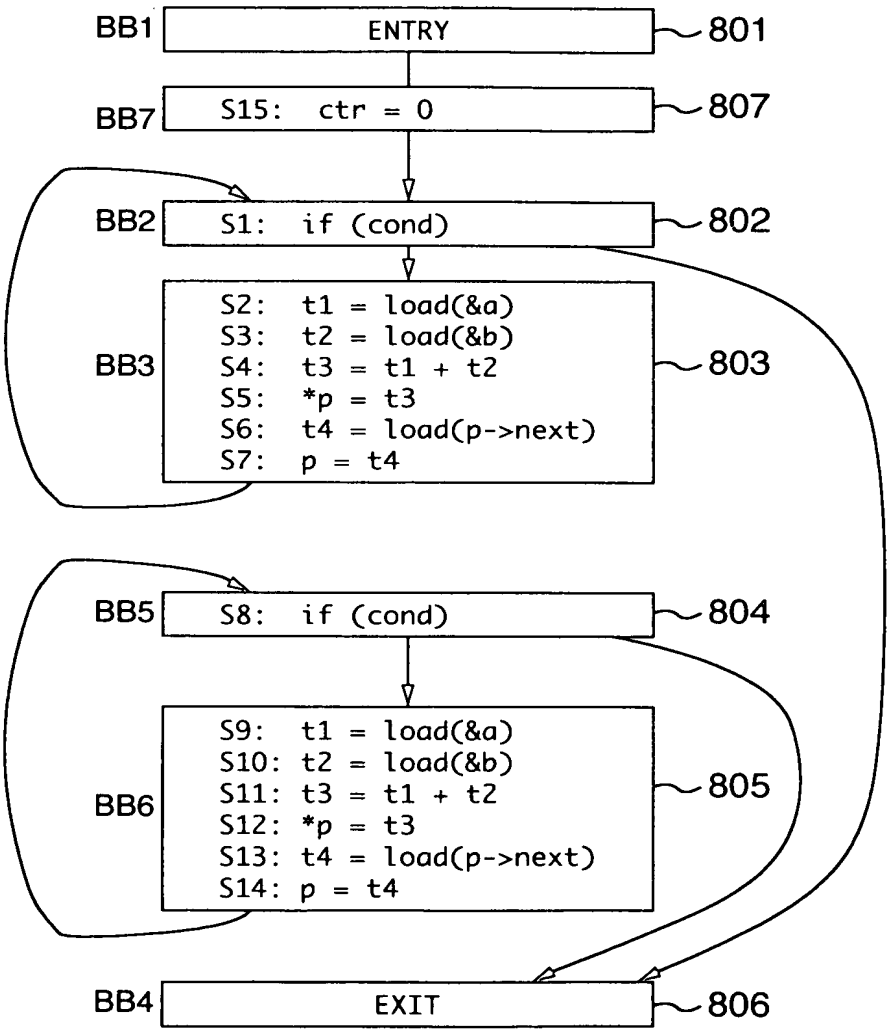
BB1
ENTRY
801
BB7
S15: ctr = 0
807
BB2
S1: if (cond)
802
BB3
S2: t1 = load(&a)
S3: t2 = load(&b)
S4: t3 = t1 + t2
S5: *p = t3
S6: t4 = load(p->next)
S7: p = t4
803
BB5
S8: if (cond)
804
BB6
S9: t1 = load(&a)
S10: t2 = load(&b)
S11: t3 = t1 + t2
S12: *p = t3
S13: t4 = load(p->next)
S14: p = t4
805
BB4
EXIT
806

FIG. 9

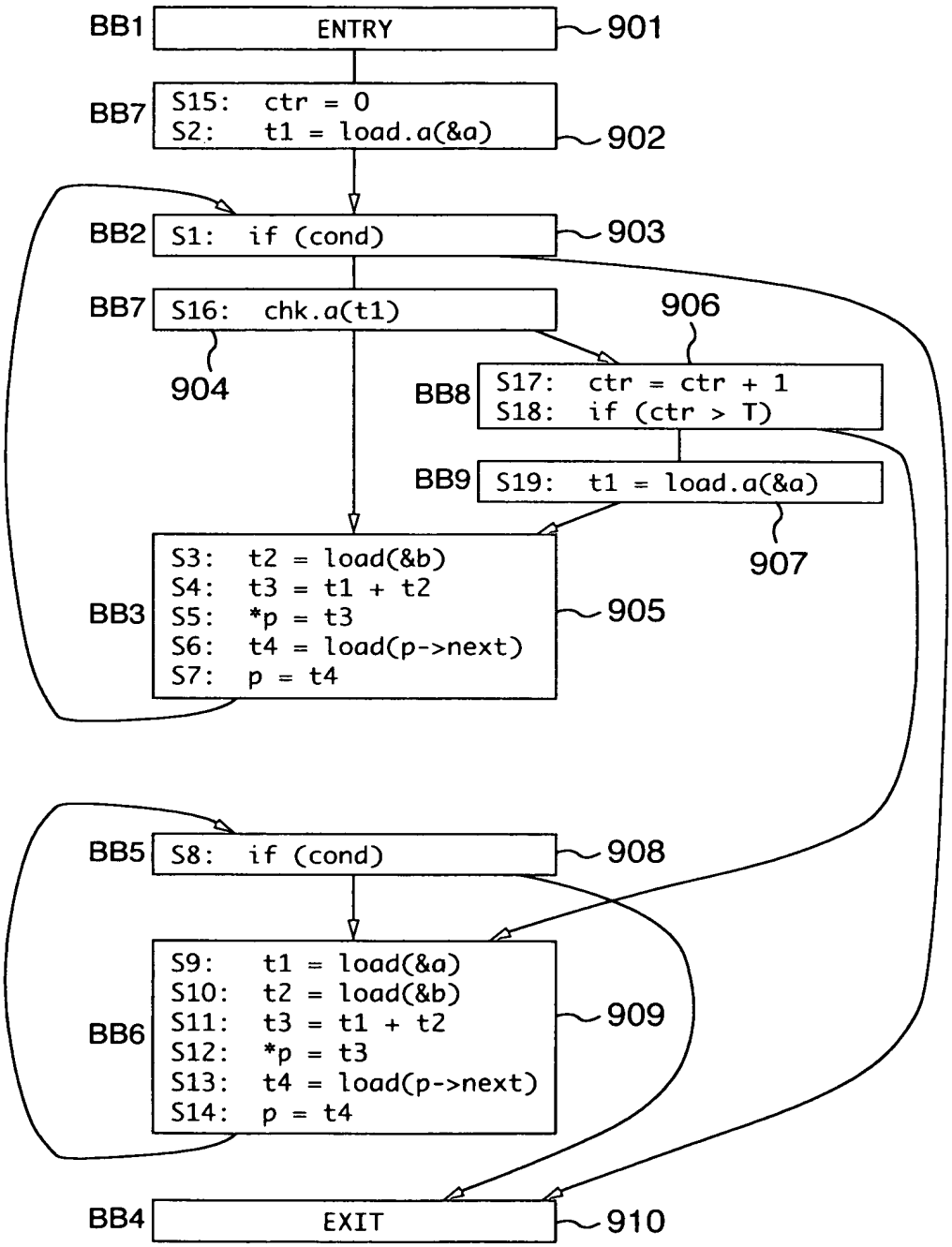
BB1
ENTRY
901
BB7
S15: ctr = 0
S2: t1 = load.a(&a)
902
BB2
S1: if (cond)
903
BB7
S16: chk.a(t1)
904
906
BB8
S17: ctr = ctr + 1
S18: if (ctr > T)
BB9
S19: t1 = load.a(&a)
907
BB3
S3: t2 = load(&b)
S4: t3 = t1 + t2
S5: *p = t3
S6: t4 = load(p->next)
S7: p = t4
905
BB5
S8: if (cond)
908
BB6
S9: t1 = load(&a)
S10: t2 = load(&b)
S11: t3 = t1 + t2
S12: *p = t3
S13: t4 = load(p->next)
S14: p = t4
909
BB4
EXIT
910

FIG. 10

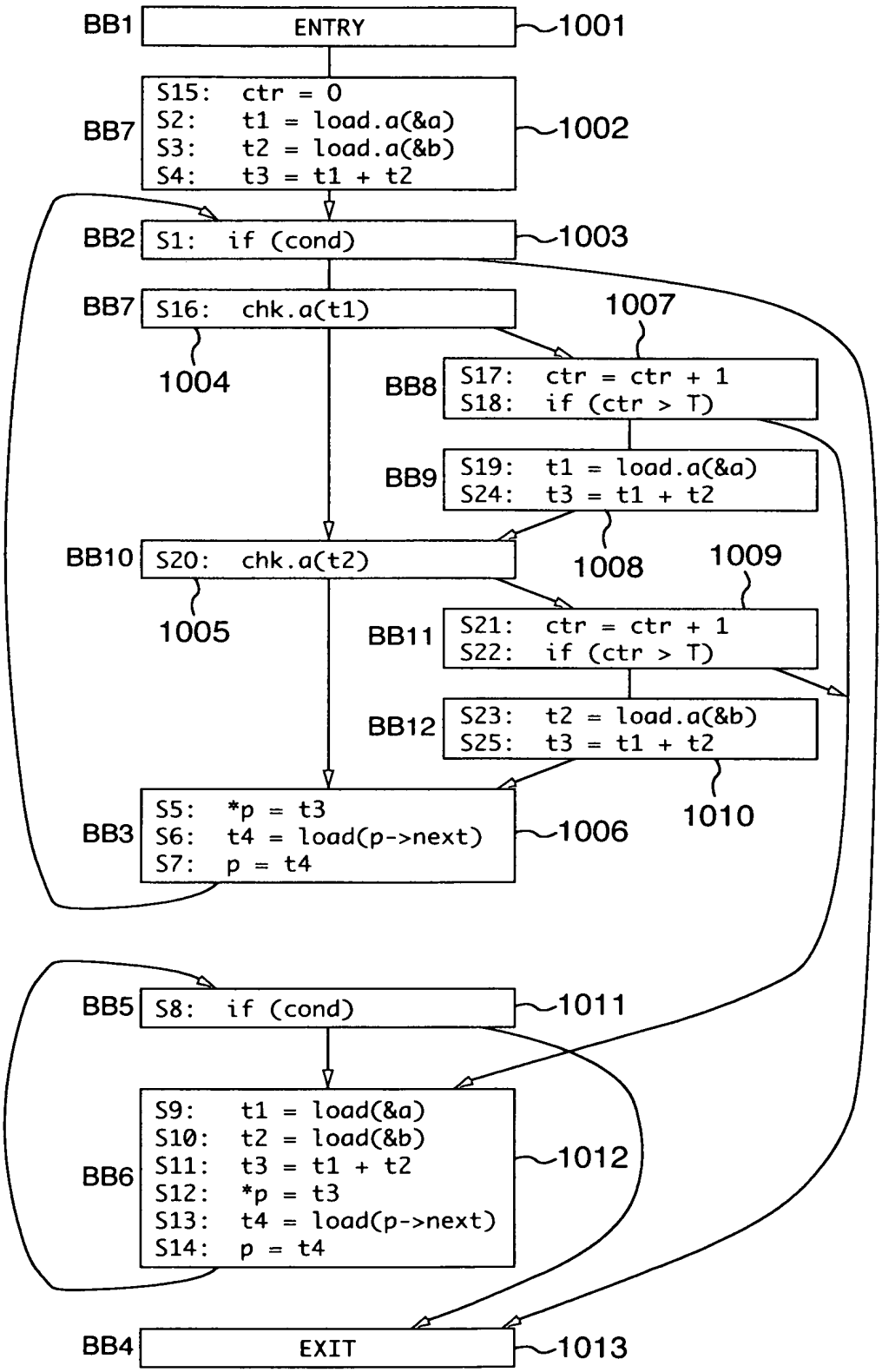
BB1
ENTRY
1001
BB7
S15: ctr = 0
S2: t1 = load.a(&a)
S3: t2 = load.a(&b)
S4: t3 = t1 + t2
1002
BB2
S1: if (cond)
1003
BB7
S16: chk.a(t1)
1004
1007
BB8
S17: ctr = ctr + 1
S18: if (ctr > T)
BB9
S19: t1 = load.a(&a)
S24: t3 = t1 + t2
1008
BB10
S20: chk.a(t2)
1005
1009
BB11
S21: ctr = ctr + 1
S22: if (ctr > T)
BB12
S23: t2 = load.a(&b)
S25: t3 = t1 + t2
1010
BB3
S5: *p = t3
S6: t4 = load(p->next)
S7: p = t4
1006
BB5
S8: if (cond)
1011
BB6
S9: t1 = load(&a)
S10: t2 = load(&b)
S11: t3 = t1 + t2
S12: *p = t3
S13: t4 = load(p->next)
S14: p = t4
1012
BB4
EXIT
1013

FIG. 11

1100

```
copy  r4=0                  (1101)
ld.a  r1=[&a]               (1102)
ld.a  r2=[&b]               (1103)
add   r3=r1,r2              (1104)
while (cond) {              (1105)
    chk.a  r1,recover1      (1106)
L1: chk.a  r2,recover2      (1107)
L2: st     *p=r3            (1108)
    p = p->next             (1109)
}                           (1110)
goto exit                   (1111)

while (cond) {              (1112)
L3: ld   r1=[&a]            (1113)
    ld   r2=[&b]            (1114)
    add  r3=r1,r2           (1115)
    st    *p=r3             (1116)
    p = p->next             (1117)
}                           (1118)
exit:                       (1119)

-----------------------------------

recover1:                   (1120)
add   r4=r4,1               (1121)
bc    r4>T,L3               (1122)
ld.a  r1=[&a]               (1123)
add   r3=r1,r2              (1124)
br    L1                    (1125)

-----------------------------------

recover2:                   (1126)
add   r4=r4,1               (1127)
bc    r4>T,L3               (1128)
ld.a  r2=[&b]               (1129)
add   r3=r1,r2              (1130)
br    L2                    (1131)
```

FIG. 12

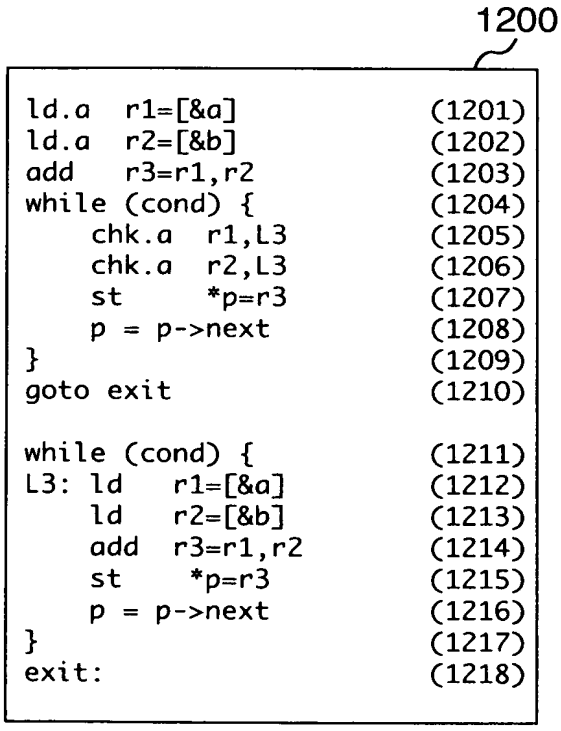
1200
ld.a r1=[&a] (1201)
ld.a r2=[&b] (1202)
add r3=r1,r2 (1203)
while (cond) { (1204)
chk.a r1,L3 (1205)
chk.a r2,L3 (1206)
st *p=r3 (1207)
p = p->next (1208)
} (1209)
goto exit (1210)
while (cond) { (1211)
L3: ld r1=[&a] (1212)
ld r2=[&b] (1213)
add r3=r1,r2 (1214)
st *p=r3 (1215)
p = p->next (1216)
} (1217)
exit: (1218)

FIG. 13

1300

```
copy  r4=0                    (1301)
copy  r5=0                    (1302)
ld.a  r1=[&a]                 (1303)
ld.a  r2=[&b]                 (1304)
add   r3=r1,r2                (1305)
while (cond) {                (1306)
    add    r5=r5,1            (1307)
    chk.a  r1,recover1        (1308)
L1: chk.a  r2,recover2        (1309)
L2: st     *p=r3              (1310)
    p = p->next               (1311)
}                             (1312)
goto exit                     (1314)

while (cond) {                (1315)
L3: ld   r1=[&a]              (1316)
    ld   r2=[&b]              (1317)
    add  r3=r1,r2             (1318)
    st    *p=r3               (1319)
    p = p->next               (1320)
}                             (1321)
exit:                         (1322)

-------------------------------------

recover1:                     (1323)
add   r4=r4,1                 (1324)
div   r6=r4,r5                (1325)
bc    r6>T,L3                 (1326)
ld.a  r1=[&a]                 (1327)
add   r3=r1,r2                (1328)
br    L1                      (1329)

-------------------------------------

recover2:                     (1330)
add   r4=r4,1                 (1331)
div   r6=r4,r5                (1332)
bc    r6>T,L3                 (1333)
ld.a  r2=[&b]                 (1334)
add   r3=r1,r2                (1335)
br    L2                      (1336)
```